2IC9

EXPRESS MAIL MAILING LABEL NO. EV 318617335 US

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:
Uresh K. Vahalia & Percy Tzelnic

Serial No.: 09/261,621

Filed: March 3, 1999

For: File Server System Providing Direct Data Sharing Between Clients With A Server Acting As An Arbiter and Coordinator

Group Art Unit: 2154

Examiner: Dustin Nguyen

Atty. Dkt. No.: EMCR:039

APPEAL BRIEF TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

#18
Turner B
6/03/04
2nd 1/83

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

Please find enclosed this original and two copies of this Appeal Brief to the Board of Patent Appeals and Interferences. Please deduct the $ 330 fee of 37 C.F.R. 1.17(c) for filing a brief in support of the appeal from EMC Corporation Deposit Account No. 05-0889, Order No. EMC-98-092. A Fee transmittal form is enclosed for this purpose.

I. REAL PARTY IN INTEREST

The real party in interest is EMC Corporation, by virtue of an assignment recorded at Reel 9803 Frame 0548.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF THE CLAIMS

Claims 1 to 50 have been presented for examination.

Claims 1 to 50 have been finally rejected, and are being appealed.

IV. STATUS OF AMENDMENTS

An amendment was filed on Feb. 26, 2004 after the final Official Action, and according to an Advisory Action mailed March 18, 2004, this amendment has not been entered.

V. SUMMARY OF THE INVENTION

The invention relates generally to data storage systems, and more particularly to network file servers. (Appellants' specification, page 1, lines 2-4.) In particular, data consistency problems may arise if concurrent client access to a read/write file is permitted through more than one data mover. (Appellants' specification, page 3, lines 1-2.)

The invention provides a method of operating a file server in a data network. The file server receives a request for metadata about a file to be accessed. The request is received from a data processing device in the data network. In response to the request for metadata, the file server grants to the data processing device a lock on at least a portion of the file, and returns to the data processing device metadata of the file including information specifying data storage locations in the file server for storing data of the file. (Claim 1.)

For example, as shown in appellants' FIG. 3, as reproduced below, before reading or writing to the file system 62, a client first issues a request for metadata to the data mover 61.

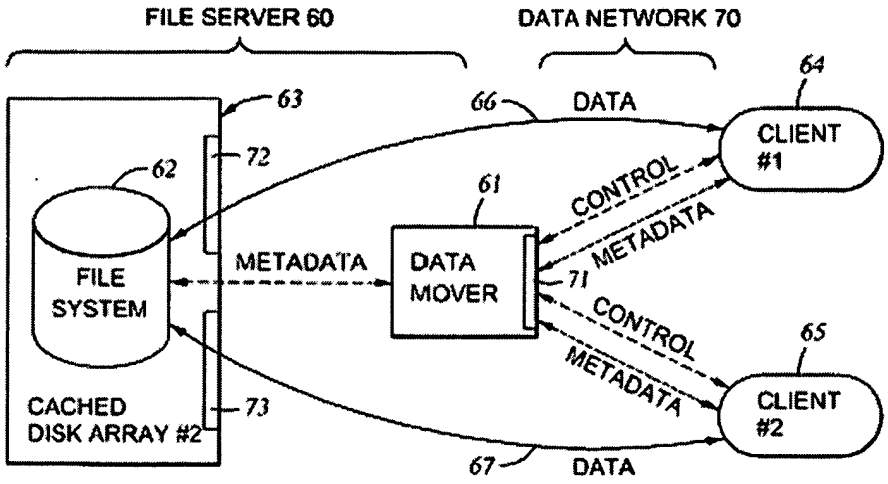
FILE SERVER 60
DATA NETWORK 70
63
66
DATA
64
CLIENT #1
72
62
61
CONTROL
METADATA
FILE SYSTEM
METADATA
DATA MOVER
71
CONTROL
METADATA
65
CLIENT #2
CACHED DISK ARRAY #2
73
67
DATA

Fig. 3

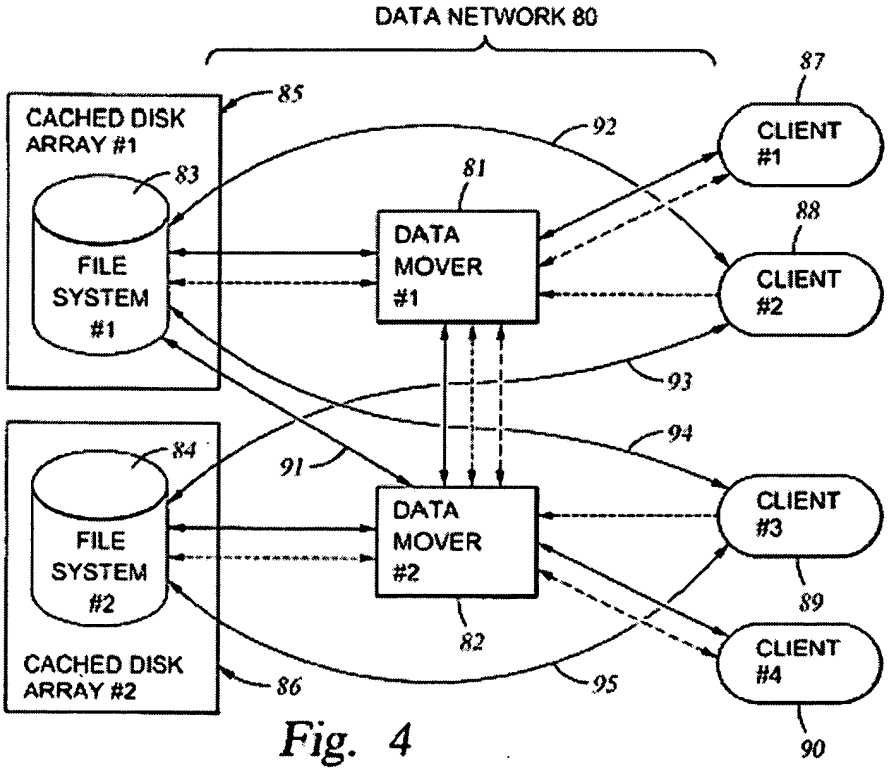
DATA NETWORK 80
87
85
CACHED DISK ARRAY #1
92
CLIENT #1
83
81
88
DATA MOVER #1
FILE SYSTEM #1
CLIENT #2
93
94
84
91
DATA MOVER #2
CLIENT #3
FILE SYSTEM #2
89
82
CACHED DISK ARRAY #2
86
95
CLIENT #4
90

Fig. 4

The data mover 61 responds by placing an appropriate lock on the file to be accessed, and returning metadata including pointers to where the data to be accessed is stored in the file system. The client uses the metadata to formulate a read or write request sent over the bypass data path to the file system 62. If the write request changes the file attributes, then the client writes the new file attributes to the data mover 61 after the data is written to the file system 62. (Appellants' specification, page 14, lines 18-25.)

In another example, shown in FIG. 4, as reproduced above, a second client 88 accesses the first file system 83 in the fashion described above with reference to FIG. 3, and a third client 89 accesses the second file system 84 in the fashion described above with reference to FIG. 3. For example, to access the first file system 83, the second client 88 sends a metadata request to the first data mover 81. The first data mover 81 places a lock on the file to be accessed, and returns metadata including pointers to the data in the file to be accessed. The second client 88 uses the pointers to formulate a corresponding data access command sent over the bypass data path 92 to the first file system 83, and any read or write data is also communicated over the bypass data path 92 between the first file system 83 and the second client 88. (Appellants' specification, page 16, lines 7-15.)

The file server 60 in FIG. 3 provides direct data sharing between network clients 64, 65 by arbitrating and coordinating data access requests. The data mover 61 grants file lock request from the clients 64, 65 and also provides metadata to the clients 64, 65 so that the clients can access data storage 62 in the cached disk array 63 over a data path that bypasses the data mover

61. The data mover 81, 82 and the clients 88, 89 in FIG. 4 may operate in a similar fashion. (Appellants' specification, page 55, lines 18-24.)

The network file server architecture of FIG. 4 allows file sharing among heterogeneous clients, and supports multiple file access protocols concurrently. The architecture permits clients using traditional file access protocols to inter-operate with clients using the new distributed locking and metadata management protocol for direct data access at the channel speed of the data storage devices. This provides a scaleable solution for full file system functionality for coexisting large and small files. (Appellants' specification, page 58, line 30, to page 59, line 5.)

VI. ISSUES ON APPEAL

1. Whether claims 1-4, 6-8, 11-15, 19-20, 23, 25-27, 30-35, 36-45, and 49 are unpatentable under 35 U.S.C. 103(a) over St. John Herbert, III (U.S. Patent No. 6,366,917) in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624).

2. Whether claims 5, 16-18, 22, 24, and 46-48 are unpatentable under 35 U.S.C. 103(a) over St. John Hubert, III (U.S. Patent No. 6,366,917), in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624) and Galeazzi et al. (U.S. Patent No. 6,535,868).

3. Whether claims 9, 10, 21, 28, 29, and 50 are unpatentable under 35 U.S.C. 103(a) over St. John Herbert, III (U.S. Patent No. 6,366,917) in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624) and Goldberg et al. (U.S. Patent No. 6,076,092).

VII. GROUPING OF THE CLAIMS

Group I: Claims 1-4, 6-8, 11-15, 19-20, 23, 25-27, 30-35, 36-45, and 49.

Group II: Claim 5, 16-18, 22, 24, and 46-48.

Group III: Claims 9, 10, 21, 28, 29, and 50.

Appellants state that the claims in Group I do not stand or fall together.

Appellants state that the claims in Group II do not stand or fall together.

Appellants state that the claims in Group III do not stand or fall together.

VIII. ARGUMENT

1. Claims 1-4, 6-8, 11-15, 19-20, 23, 25-27, 30-35, 36-45, and 49 are patentable under 35 U.S.C. 103(a) over St. John Herbert, III (U.S. Patent No. 6,366,917) in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624).

On page 10 of the Final Official Action, claims 1-4, 6-8, 11-15, 23, 25-27, 30-35, and 36-45 were rejected under 35 U.S.C. 103(a) as being unpatentable over St. John Herbert, III (U.S. Patent No. 6,366,917) in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624). (Claims 19 and 20 were rejected on page 15 for the same reasons as claims 6 and 7, and claim 49 was rejected on page 17 for the same reasons as claim 6.) Appellants respectfully traverse.

The policy of the Patent and Trademark Office has been to follow in each and every case the standard of patentability enunciated by the Supreme Court in Graham v. John Deere Co., 148 U.S.P.Q. 459 (1966). M.P.E.P. § 2141. As stated by the Supreme Court:

> Under § 103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background, the obviousness or nonobviousness of the subject matter is determined. Such secondary considerations as commercial success, long felt but unsolved needs, failure of others, etc., might be utilized to give light to the circumstances surrounding the origin of the subject matter sought to be patented. As indicia of obviousness or nonobviousness, these inquiries may have relevancy.

148 U.S.P.Q. at 467.

The problem that the inventor is trying to solve must be considered in determining whether or not the invention would have been obvious. The invention as a whole embraces the structure, properties and problems it solves. In re Wright, 848 F.2d 1216, 1219, 6 U.S.P.Q.2d 1959, 1961 (Fed. Cir. 1988).

Paragraph 37 on page 10 of the Final Official Action cites St. John Herbert, III for disclosing "a method of operating a file server in a data network, said method comprising: the file server receiving a request for metadata about a file to be accessed [col 3, lines 9-20 and lines 34-38], the request being received from a data processing device in the data network [22, 24, 26, Figure 1B]." However, the server in St. John Herbert, III provides access to a generated database in a database system (e.g., an Oracle 8 product from Oracle Corporation) and to metadata in a metadata database. [Col. 3 lines 18-32.] The metadata comprises a collection of metadata objects in the form of tables (metatables), which describe the structure of objects within the generated database. "For example, the metadata objects may describe the various columns that constitute a table within the generated database 18, and may further specify the format, type or

other characteristics of these columns. The metadata objects may also include management information pertaining to tables and columns of the generated database 18, such as description information." [Col. 3, lines 49-60.]

The Final Official Action further says that St. John Herbert does not specifically disclose "in response to the request for metadata, the file server granting to the data processing device a lock on at least a portion of the file, and returning to the data processing device metadata of the file including information specifying data storage location in the file server for storing data of the file."

The Final Official Action cites Schmuck for disclosing "in response to the request for metadata [col. 2, lines 57-59], the file server granting to the data processing device a lock on at least a portion of the file [col 34, lines 64-col 35, lines 9]." However, these two cited passages in Schmuck are dealing with different data processing systems, so it is not understood how the granting of the lock on at least a portion of a file in the second passage is done "in response to" the request for metadata of the first passage. The first passage is from the "Background of the Inventions" in Schmuck, col. 2, lines 52-59, which says:

> Represented by what may be the best of breed for Audio/Video file systems (IBM's VideoCharger Server for AIX), previous solutions dealing with computer systems which would allow standards compliance have relied on shipping file system level requests to a single server which acquires the data and returns it or shipping metadata requests from a client to a single server which allows the original computer to directly fetch the data.

The second passage of Schmuck [col 34, lines 64-col 35, lines 9] relates to a shared disk file system in FIG. 1 of Schmuck, which includes a shared token manager for nodes of the computer system. Each node (1, 2, 3) in this system is a respective computer having a respective file system and shared access to disks. [See Schmuck, col. 5, lines 24-27.] "There is no file system server in the path for either data or metadata. Any available path can be used avoiding a server as a bottleneck or as a single point of failure. Since the required central functions in the lock manager have no attachment to a specific computer, they can be migrated from computer to computer to satisfy performance and availability needs." [Schmuck col. 5, lines 42-47.] The use of a token in the system of FIG. 1 is described in Schmuck, col. 34, lines 64-col. 35, line 9:

> Serializing accesses to different regions in a file to which processes on different nodes write in parallel is done by distributed byte range locks. When a process needs to lock a byte range, it first needs to acquire an appropriate byte range token. The byte range token represents the node's access rights to a portion of a file. Thus, if a node holds a byte range token for file X for range (100, 200) in read mode, it means that the node may safely read that portion of the file. However, to prevent stealing the token, the node must lock the token before the actual read, since if another node needs to write the same portion, it might steal the token. Locking the token prevents the steal. After the read has completed, the token is unlocked.

Page 10, paragraph 37 of the Final Official Action says Teare discloses "returning to the data processing device metadata of the file including information specifying data storage location in the file server for storing data of the file [col 2, lines 4-10; and col. 4, lines 54-57]." Page 3,

paragraph 8 of the Final Official Action further cites Teare col. 2, lines 17-24 and col. 5, lines 17-27. However, col. 2 lines 4-24 of Teare say:

> Each item of information available using the Web, including files, images, or pages, is called a resource. A Uniform Resource Locator (URL) uniquely identifies each resource stored on a server. A URL is a form of network address comprising a domain name coupled to an identifier of the location of information stored in a network.
>
> An example of a URL is http://www.centraal.com/index.html. In this example, "http://" indicates that the information associated with the URL can be accessed using HTTP; www.centraal.com identifies the server that is storing the information; and "index.html" identifies a file or page on that server.
>
> The local computer requests information by providing a request containing URL of the desired information to the remote server. The server receives the request, locates the page of information corresponding to the URL, and returns the page to the local computer over the HTTP connection. The pages of information are files prepared in the Hypertext Markup Language (HTML). The local computer runs a browser program that can read HTML files, interpret HTML codes in the files, and generate a complex graphical display.

Column 4, lines 54-57 of Teare et al. are part of a paragraph in the "Summary of the Invention" that says:

> The foregoing needs, and other needs and objects, are fulfilled by the present invention, which comprises, in one aspect, a method of navigating, based upon a

> natural language name, to a resource that is stored in a network and identified by a location identifier, comprising the steps of storing a first natural language name of the resource in association with the location identifier of the resource; receiving a request to locate the resource containing the first natural language name; retrieving the location identifier associated with the first natural language name; and delivering the resource to the user using the location identifier.

Col. 5, lines 17-27 of Teare say:

> According to another feature, the method includes the steps of receiving a client identifier of a client associated with the resource; generating a set of metadata that describes the resource, the location identifier, and the client identifier; and storing the set of metadata in a persistent storage device associated with the client.

With respect to Teare, the Final Official Action appears to equate the Uniform Resource Locator (URL) of a resource stored on a server in the World Wide Web with "information specifying data storage locations in the file server for storing data of the file." However, the example in col. 2, lines 11-16 in Teare illustrates that the URL of a file on the Web server does not specify data storage locations in the file server for storing data of the file. The URL http://www.centraal.com/index.html has a portion www.centraal.com that identifies the server that is storing the information, and "index.html" that identifies the file or page on that server. However, there is nothing in the URL that specifies data storage locations in the file server for storing data of the file. The limit of specificity here is "index.html", which is essentially a name of the file in the

server. The data storage locations for storing data of the file would be known to the server but they are not specified by "index.html".

In contrast, the appellants' specification provides support for "information specifying data storage locations in the file server for storing data of the file" by describing that the appellants' file server provides a user with "pointers to where the data to be accessed is stored in the file system." (Page 14, lines 20-22.) This permits the client to use the pointers to "formulate a read or write request sent over the bypass data path to the file system 62." (Page 14, lines 22-23). For example, the pointers point "to where the file data resides in the cached disk array storing the file system." (Page 31 lines 13-14). Moreover, a file name does not specify data storage locations in the file server for storing data of the file because a primary function of the file manager in the file server is to map file names to data storage locations allocated to the files by the file manager. (See appellants' specification, page 2, lines 17-20.)

On page 11, paragraph 37 of the Final Official Action concludes: "It would have been obvious to a person skill[ed] in the art at the time the invention was made to combine St. John Herbert, Schmuck and Teare because Schmuck's locking mechanism would provide efficient basic file control in a shared disk environment for multiple computers [Schmuck, col 3, lines 50-57]." The appellants respectfully disagree. It is not seen how any proper combination of St. John Herbert, Schmuck and Teare would result in appellants' invention of claim 1. Moreover, the cited references fail to provide a proper motivation for combining St. John Herbert, Schmuck and Teare as proposed in the Final Official Action.

As discussed above, neither St. John Herbert, Schmuck, nor Teare disclose that information specifying data storage locations in the file server for storing data of the file should

be returned by the file server to a data processing device in the data network in response to a request from the data processing device to the file server for metadata about the file. Where the prior art references fail to teach a claim limitation, there must be "concrete evidence" in the record to support an obviousness rejection. "Basic knowledge" or "common sense" is insufficient. *In re Zurko*, 258 F.3d 1379, 1385-86, 59 U.S.P.Q.2d 1693, 1697 (Fed. Cir. 2001).

Nor do the purported advantages of Schmuck's locking mechanism (i.e., "efficient basic file control in a shared disk environment for multiple computers sharing the disk and file environment" in col. 3, lines 50-52) provide proper motivation for combining St. John Herbert, Schmuck and Teare as proposed in the Final Official Action. The purported advantages of Schmuck's locking mechanism are represented as attributes of Schmuck's parallel file system shown in FIG. 1 of Schmuck, so they do not provide any motivation for modifying Schmuck's parallel file system. Instead, the parallel file system of Schmuck appears to be entirely satisfactory for its intended purpose of permitting multiple computer nodes to share access to files in disk storage.

Moreover, the appellants' specification, page 17, lines 1-10, teach that use of the invention is desirable only under particular circumstances:

> Whenever a client has a bypass data path to a file system and can therefore send data access commands to the file system without passing through a data mover computer, the client can potentially access all of the files in the file system. In this situation, the client must be trusted to access only the data in a file over which the client has been granted a lock by the data mover that owns the file system to be accessed. Therefore, the methods of client access as described above with reference to FIGS. 2 and 3 have a security risk that may not be acceptable for

clients located in relatively open regions of the data network. The method of client access as described above with reference to FIG. 3 also requires special client software, in contrast to the methods of client access as described above with reference to FIGS. 1-2 which can use standard client software.

Furthermore, the Final Official Action proposes to modify the relational database system of St. John Herbert, III, in view of the discussion of audio/visual file systems in Schmuck col. 2, lines 57-59, in view of the disclosure of the locking by the token manager in the parallel file system of Schmuck FIG. 1 [col 34, lines 64-col 35, lines 9], and in view of Teare's navigation of network resources in the World Wide Web by providing a lookup service for finding the URL of a file or page in the World Wide Web given a natural language name. Thus, the cited references are dealing with four different kinds of data processing systems (database server, video file server, multiprocessor system, and a name finding service in the World Wide Web) and various different kinds of metadata. There is no apparent reason why a person of ordinary skill, seeking to improve the performance of a network file server, would be looking at such disparate references and more importantly selecting the particular portions of the different systems in these references for combination and modification to reconstruct the appellants' claimed invention.

Hindsight reconstruction, using the appellants' specification itself as a guide, is improper because it fails to consider the subject matter of the invention "as a whole" and fails to consider the invention as of the date at which the invention was made. "[T]here must be some motivation, suggestion, or teaching of the desirability of making the specific combination that was made by the applicant." In re Lee, 277 F.3d 1338, 1343, 61 U.S.P.Q.2d 1430, 1435 (Fed. Cir. 2002) (quoting In re Dance, 160 F.3d 1339, 1343, 48 U.S.P.Q.2d 1635, 1637 (Fed. Cir. 1998)).

"[T]eachings of references can be combined only if there is some suggestion or incentive to do so." In re Fine, 837 F.2d 1071, 1075, 5 U.S.P.Q.2d 1596, 1600 (Fed. Cir. 1988) (Emphasis in original) (quoting ACS Hosp. Sys., Inc. v. Montefiore Hosp., 732 F.2d 1572, 1577, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984)). "[P]articular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed." In re Kotzab, 217 F.3d 1365, 1371, 55 U.S.P.Q.2d 1313, 1317 (Fed. Cir. 2000). See, for example, Fromson v. Advance Offset Plate, Inc., 755 F.2d 1549, 1556, 225 U.S.P.Q. 26, 31 (Fed. Cir. 1985) (nothing of record plainly indicated that it would have been obvious to combine previously separate lithography steps into one process); In re Gordon et al., 733 F.2d 900, 902, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984) (mere fact that prior art could be modified by turning apparatus upside down does not make modification obvious unless prior art suggests desirability of modification); Ex Parte Kaiser, 194 U.S.P.Q. 47, 48 (PTO Bd. of Appeals 1975) (Examiner's failure to indicate anywhere in the record his reason for finding alteration of reference to be obvious militates against rejection).

With respect to claim 2, page 11, paragraph 38 of the Final Official Action recognizes that St. John Herbert does not specifically disclose any of the recited elements other than a data storage device. The Final Official Action cites Schmuck for disclosing the other recited elements in the claim. However, Schmuck [col. 31, lines 64-col 32, line 8] deals with a multiprocessor system in which:

> In this system, a node is appointed for each file which is responsible for accessing and updating the file's metadata. This metadata node (or metanode) shares this information with other nodes upon request.

> The metadata node keeps the information about the file's metadata and acts as a smart cache between the disk and all the nodes that access the file. There are situations when the metadata node (or metanode) ceases to serve this function. In order to enable smooth operation and recovery, these situation[s] need to be handled. Nodes that used to access the metanode need to elect a new metanode in a straightforward way.

However, from the cited references, it is not seen how or why metadata caching in a multi-processor system such as Schmuck (FIG. 1) should be applied to caching of metadata in a file server for providing the metadata (including the information specifying data storage locations for storing data of the file) to a data processing device (e.g., a client) in a data network. One would expect a file server caching metadata to keep to itself the information specifying data storage locations for storing data of the file. Moreover, in Schmuck (FIG. 1), the token manager (11) manages locks, and the nodes (i.e., computers 1, 2, 3) access and update the metadata.

With respect to claim 3, Schmuck col. 32, lines 15-25, further discloses that the token manager is a distributed subsystem which grants tokens to nodes. The token manager is not a file server distinct from the nodes (i.e., computers 1, 2, 3). In the appellants' claims, the file server is distinct from the data processing device or client; in particular, the file sever receives from the data processing device a request for metadata about a file to be accessed, the file server grants the lock to the data processing device, and returns to the data processing device metadata of the file including information specifying data storage locations in the file server for storing data of the file.

Paragraph 40 on page 12 of the Final Official Action says: "As per claim 4, St. John Herbert discloses the data processing device writes data to the data storage locations in the file server, modifies the metadata from the file server in accordance with the data storage locations in the file server to which the data is written [Abstract], and sends the modified metadata to the file server [154, 156, Figure 20]. Appellants respectfully disagree. St. John Herbert discloses a data base system including a database server (FIG. 1A) or a database server and metadata server (FIG. 1B). St. John Herbert is concerned with "allowing a user to specify modifications to the structure of the generated database 18 in the metadata database 16 (and specifically the metadata application dictionary 19), and then utilizing these modified specifications and relationships to update the generated database 18." (Col. 4, lines 3-7.) The metadata comprises a collection of metadata objects in the form of tables (metatables), which describe the structure of objects within the generated database. "For example, the metadata objects may describe the various columns that constitute a table within the generated database 18, and may further specify the format, type or other characteristics of these columns. The metadata objects may also include management information pertaining to tables and columns of the generated database 18, such as description information." (Col. 3, lines 49-60.) Thus, in St. John Herbert, it is not seen where a data processing device distinct from any file server writes data to the data storage locations in the file server, modifies the metadata from the file server in accordance with the data storage locations in the file server to which the data is written, and sends the modified metadata to the file server. The metadata described in col. 3, lines 49-60 of St. John Herbert does not include information specifying data storage locations for storing data of a file, nor would the metadata described in col. 3, lines 49-60 of St. John Herbert be modified in accordance with data storage locations in

the database server to which data would have been written by the data processing device or client.

In a conventional file server, it is the file server that is mapping the file name to the storage locations, and modifying the metadata of the file when needed in accordance with the data storage locations in the file server to which the data is written. Claim 4 instead defines that the data processing device is writing data to the data storage locations in the file server, and modifying the metadata from the file server in accordance with the data storage locations in the file server to which the data is written. In short, claim 4 defines that the data processing device (e.g., the client) is performing operations previously performed by the file server. In a similar fashion, in a database server, the database server would map objects in the database (such as tables and the cells in the tables) to data storage locations in the database server. There is nothing in St. John Herbert to suggest that the user or client in St. John Herbert knows or cares about the data storage locations in the database server, so that the user or client would not be able to modify any metadata from the file server "in accordance with the data storage locations in the file server to which the data is written." In short, FIG. 20 of Herbert shows a process in which the user modifies metadata via a user interface (UI), and the database server modifies tables in the database in accordance with modification to the metadata, but this metadata is not modified in accordance with data storage locations in the server to which the user or client has written data.

With respect to claim 6, paragraph 41 on pages 12-13 of the Final Official Action acknowledges that St. John Herbert does not disclose the elements expressly recited in claim 6, and cites Schmuck col. 43, lines 54-64 for these elements. Schmuck col. 43, lines 54-64 say:

> Thus, every node eventually finds out either that it has become the new metanode or that the metanode has changed. In either case, appropriate actions are taken. If a node became a metanode, it reads the most recent metadata from disk. If a node's metanode changed, the node will re-send its own metadata updates to the new metanode since its possible that the old metanode failed before flushing these updates to disk. By using a version number for each such update, every node knows which updates are on disk and which have to be re-sent to the new metanode.

Although Schmuck discloses that a version number is associated with each metadata update, it is not seen where the data processing device includes this version identifier in the request for access to the file, so that the file server compares the version identifier from the data processing device to a version identifier of a most recent version of the metadata of the file, in order to return the most recent version of metadata to the data processing device when the metadata of the file cached in the cache memory of the data processing device is not the most recent metadata of the file. Instead, it appears that the version number in Schmuck is used to maintain coherency in the multi-processor system of Schmuck between the metadata cache of the metanode and the metadata on disk, for example, when there is a crash of the old metanode and assignment of a new metanode.

Paragraph 41 on page 13 of the Final Official Action concludes: "It would have been obvious to a person skill[ed] in the art at the time that the invention was made to combine the teachings of St. John Herbert and Schmuck because Schmuck's teaching of versioning would provide an additional step for keep[ing] track of data integrity." Appellants respectfully

disagree. Even if the database server in St. John Herbert were provided with multiple processors, a metadata cache in each processor, and a version identifier associated with each metadata update, there is no suggestion that the user or client should be provided with a metadata cache, and the user or client should include the version identifier of its cached metadata in a request for access to the database in the database server, in order for the database server to return the most recent version of metadata to the user or client when the metadata in the cache memory of the user or client would not be the most recent metadata of the file.

Paragraph 43 on page 14 of the Final Official Action rejected claim 8 for similar reasons as claim 1. Appellants respectfully traverse, for the same reasons as discussed above with respect to claim 1. In addition, claim 8 further defines the operation of the client in the data processing network. The file server sends to the client metadata of the file including information specifying data storage locations in the file server for storing data of the file, and the client uses this metadata of the file for producing at least one data access command for accessing the data storage locations in the file server, and sends the data access command to the file server to access the data storage locations of the file server. This further distinguishes Schmuck because there is no differentiation of the computer nodes and disks in the multiprocessor system of FIG. 1 of Schmuck into a file server and a client in a data network.

The Final Official Action further cites Teare. In Teare, however, the user requests metadata (a URL) from a lookup service in the World-Wide Web. The URL specifies a web site (e.g., a file server) and the name of a file or page stored at the web site. The URL does not specify the data storage locations of the file or page stored at the web site. The user then uses the

URL lookup service to access the web site to retrieve the file or page specified by the URL from the web site.

With respect to claim 11, paragraph 44 on page 14 of the Final Official Action rejected claim 11 for similar reasons as stated above in claim 3. Appellants respectfully traverse, for the reasons given above with respect to claim 3.

With respect to claim 14, paragraph 47 on page 15 of the Final Official Action rejects claim 14 for similar reasons as stated above in claim 8. Appellants respectfully traverse, for the reasons given above with respect to claim 8, and more importantly because claim 8 defines the particular input-output related operating system routines that are dynamically linked to the application programs of the client. It is not seen where the cited references suggest that these input-output related operating system routines of the client should obtain from the file server locks upon at least a portion of each of the files, obtain metadata for producing data access commands for accessing data storage in the file server, produce the data access commands from the metadata, and send the data access commands to the file server in order to access the data storage of the file server. The claimed input-output routines have the effect of transferring and incorporating into the operating system level of the client various functions that were previously performed by the network file server.

Paragraph 47 of the Final Official Action cites col. 3, lines 20-25 of St. John Herbert for disclosing dynamically linking application programs of the client with input-output related operating system routines of the client. This passage says:

> In one embodiment of the invention, each of the clients 22, 26 and 30 comprises a browser, such as the Navigator browser from Netscape Communications Corp. of

Mountain View, California or the Internet Explorer (IE) browser from Microsoft Corp. of Redmond, Washington State.

This passage of St. John Herbert fails to suggest that input-output related operating system routines of the client should obtain from the file server locks upon at least a portion of each of the files, obtain metadata for producing data access commands for accessing data storage in the file server, produce the data access commands from the metadata, and send the data access commands to the file server in order to access the data storage of the file server.

Paragraph 48 on page 15 of the Final Official Action rejected claim 15 for similar reasons as stated above with respect to claim 4. Appellants respectfully traverse, for the reasons as stated above with respect to claim 4.

Paragraph 49 on page 15 of the Final Official Action rejected claim 19 for similar reasons as stated above with respect to claim 6. Appellants respectfully traverse, for the reasons as stated above with respect to claim 6.

Paragraph 50 on page 16 of the Final Official Action rejected claims 23 and 25 for similar reasons as stated above with respect to claims 2 and 6. Appellants respectfully traverse, for the reasons as stated above with respect to claims 2 and 6.

Paragraph 51 on page 16 of the Final Official Action rejected claim 27 for similar reasons as stated above with respect to claims 1 and 8. Appellants respectfully traverse, for the reasons as stated above with respect to claims 1 and 8.

Paragraph 53 on page 16 of the Final Official Action rejected claim and 34 for similar reasons as stated above with respect to claim 15. Appellants respectfully traverse, for the reasons as stated above with respect to claim 15.

Paragraph 54 on page 16 of the Final Official Action rejected claim 35 for similar reasons as stated above with respect to claim 6. Appellants respectfully traverse, for the reasons as stated above with respect to claim 6.

Paragraph 55 on page 16 of the Final Official Action rejected claim 36 for similar reasons as stated above with respect to claim 1. Appellants respectfully traverse, for the reasons as stated above with respect to claim 1.

Paragraph 56 on page 16 of the Final Official Action rejected claim 37 for similar reasons as stated above with respect to claim 4. Appellants respectfully traverse, for the reasons as stated above with respect to claim 4.

Paragraph 57 on page 17 of the Final Official Action rejected claims 38 and 39 for similar reasons as stated above with respect to claim 2. Appellants respectfully traverse, for the reasons as stated above with respect to claim 2.

Paragraph 58 on page 17 of the Final Official Action rejected claim 40 for similar reasons as stated above with respect to claim 6. Appellants respectfully traverse, for the reasons as stated above with respect to claim 6.

Paragraph 59 on page 17 of the Final Official Action rejected claim 42 for similar reasons as stated above with respect to claim 8. Appellants respectfully traverse, for the reasons as stated above with respect to claim 8.

Paragraph 60 on page 17 of the Final Official Action rejected claims 44-45 for similar reasons as stated above with respect to claim 14-15. Appellants respectfully traverse, for the reasons as stated above with respect to claims 14-15.

Paragraph 61 on page 17 of the Final Official Action rejected claim 49 for similar reasons as stated above with respect to claim 6. Appellants respectfully traverse, for the reasons as stated above with respect to claim 6.

2. Claims 5, 16-18, 22, 24, and 46-48 are patentable under 35 U.S.C. 103(a) over St. John Hubert, III (U.S. Patent No. 6,366,917), in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624) and Galeazzi et al. (U.S. Patent No. 6,535,868).

On page 17, paragraph 29 of the Final Official Action rejected claims 5, 16-18, 22, 24, and 46-48 under 35 U.S.C. 103(a) as being unpatentable over St. John Hubert, III (U.S. Patent No. 6,366,917), in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624) and Galeazzi et al. (U.S. Patent No. 6,535,868). Appellants respectfully traverse. St. John Hubert, Schmuck, and Teare have been distinguished above with respect to the base claims. Galeazzi fails to provide the elements and motivation missing from St. John Hubert, Schmuck, and Teare.

With respect to claim 5, paragraph 63 on pages 17 to 18 of the Final Official Action said:

> As per claim 5, St. John Herbert, Schmuck, Teare do not specifically disclose the data processing device sends the modified metadata to the file server after the data processing device writes the data to the data storage of the file server. Galeazzi discloses the data processing device sends the modified metadata to the file server after the data processing device writes the data to the data storage of the file server [col 5, lines 58-col 6, lines 15]. It would have been obvious to a person skill[ed] in the art at the time the invention was made to

combine the teaching of St. John Herbert, Schmuck, Teare and Galeazzi because Galeazzi's teaching would allow for data to be maintained for up to date information.

Galeazzi et al. is directed to a method and apparatus for managing metadata in a database management system. (Title.) The metadata is managed by accepting a command in the database management system that implicates a metadata modification and providing that command to a metadata service, parsing the command to determine if the command requires a metadata modification, accessing a rule set defining rules for modifying the metadata when indicated by the parsed command, and enforcing the rule defined in the rule set. (Abstract.) Metadata in the context of Galeazzi is "a class of data which describes characteristics of other data in the database management system." Col. 1, lines 20-23. "In a centralized data warehouse, the metadata for any given application software component consists of three parts: (1) definitions that are meaningful only to the component itself, (2) definitions that are usable by a set of components different from the one which registered the definition but which are defined in terms of base definitions that affect a still larger set of components, and (3) base definitions that must be considered a system-wide common resource." Col. 1, lines 24-34. Table 1 in col. 4, lines 1 to 11 gives examples of metadata including measures, formulas, models for multi-dimensional (hypercube) display, models for time-series analysis, and TERADATA metadata. A metadata service may receive a command in a data definition language to change the schema of the database, alter a table, or alter a column of a table. Col. 4, lines 28-33. Galeazzi et al. col. 5, line 58 to col. 6, line 8 describes a two-phase commit protocol for committing metadata changes.

In a prepare phase, the metadata service (MDS) writes a transaction identification and an MDS state type to a log. In a commit phase, the metadata is updated.

Appellants' claim 5 is dependent on claim 4, which defines that the "modified metadata" is metadata, from the file server, that has been modified by the data processor in accordance with the data storage locations in the file server to which the data is written. The updated metadata in col. 4, lines 28-33 of Galeazzi is not metadata that has been modified in accordance with data storage locations in the log to which the transaction identification and MDS state type have been written. Apparently the updated metadata in Galeazzi is not metadata that has been modified in accordance with any data storage locations to which any data has been written. Moreover, the transaction identification and MDS state type appear to be metadata about the update being made to the metadata, rather than any data to which the updated metadata is related.

It is not understood why a person of ordinary skill would combine Galeazzi with St. John Herbert, Schmuck, Teare, and Galeazzi, or how any proper combination would result in the subject matter of appellants' claim 5. Galeazzi is concerned with metadata about objects such as tables in a relational database, and not metadata about a file, such as metadata specifying data storage locations where file data is written. Nor is it seen how the two-phase protocol for committing metadata changes in the relational database of Galeazzi would suggest that a client writing data to storage in a file server should modify metadata from the file server in accordance with the data storage locations in the file server to which the data is written, and send the modified metadata to the file server after the data processing device writes the data to the data storage of the file server.

Paragraph 64 on page 18 of the Final Official Action rejected claim 16 for similar reasons as stated above with respect to claim 5. Appellants respectfully traverse, for the reasons as stated above with respect to claim 5.

With respect to claim 17, paragraph 65 on page 18 of the Final Official Action said:

> As per claim 17, St. John Herbert, Schmuck and Teare do not disclose the client performs asynchronous write operations upon the data storage locations of the file server, and wherein the client sends the modified metadata to the file server in response to a commit request from an application process of the client [col. 6, lines 24-41]. It would have been obvious to a person skill[ed] in the art at the time the invention was made to combine the teaching of St. John Herbert, Schmuck, Teare and Galeazzi because Galeazzi's teaching would provide a step for maintaining data integrity inside communication network.

However, col. 6, lines 24-41 of Galeazzi say:

> FIG. 8 is a diagram summarizing the two phase commit protocol and the message transfers between the replication service 108 and the MDS 110. One phase commit processing may be implemented as well. In this embodiment, the replication service 108 changes to a prepare state and sends a commit message to the MDS 110. Then, the MDS 110 writes the transaction identification and its state type to a log. The MDS 110 then parses the DDL and determines the appropriate changes to be made to the metadata repository, and invokes the integrity routine to determine if the changes will be allowed. If the changes do not violate integrity, then the MDS 110 sends a commit message back to the replication service 108. If the changes violate integrity, the MDS 110 sends an

> abort message back to the replication service 108. Then, the replication service 108 commits or aborts (depending on the outcome of the integrity check), before sending an acknowledgement.

In other words, Galeazzi is describing a commit of modified metadata. It is not seen where Galeazzi describes asynchronous write operations to data storage locations in the database, followed by a commit of metadata that has been modified in accordance with a write operation upon the database. Moreover, since Galeazzi is dealing with metadata about the format of tables in a relational database and not metadata about data storage locations in a file server, one of ordinary skill would not have been motivated to combine Galeazzi with the other references to reconstruct the appellants' file server.

With respect to claim 18, paragraph 66 on page 18 of the Final Official Action said: "As per claim 18, Schmuck discloses the client performs asynchronous write operations upon the data storage locations of the file server, and wherein the client sends the modified metadata to the file server when the client requests the file server to close the file [col 43, lines 37-38]."

Schmuck col. 43 lines 37-38 say that a metanode ceases to operate as a metanode when the metanode closes the file or flushes it from its cache. Schmuck says that this is asynchronous in the sense that "other nodes are not aware of this immediately." (Schmuck, col. 43, lines 31-39.)

Appellants recognize that file servers have performed asynchronous write operations and have flushed metadata for the asynchronous write operations from the metadata cache of the file server to disk storage when a file is closed. However, claim 18 defines that the client performs asynchronous write operations upon the data storage locations of the file server, and the client

sends the modified metadata to the file server (rather than a storage controller of the file server sending the metadata to disk storage) when the client requests the file server to close the file. It is not seen how the metadata node of the multi-processor system of Schmuck can be considered a client in a data processing network including a client and a file server. Instead, the metadata node appears to manage file metadata and send cached metadata to disk in a similar way that a storage controller in a file server has managed metadata and has sent cached metadata to disk.

Paragraph 67 on page 19 of the Final Official Action rejected claim 22 for similar reasons as stated above with respect to claim 5. Appellants respectfully traverse, for the reasons as stated above with respect to claim 5.

Paragraph 68 on page 19 of the Final Official Action rejected claim 24 for similar reasons as stated above with respect to claims 2 and 5. Appellants respectfully traverse, for the reasons as stated above with respect to claims 2 and 5.

Paragraph 69 on page 19 of the Final Official Action rejected claims 46-48 for similar reasons as stated above with respect to claims 16-18. Appellants respectfully traverse, for the reasons as stated above with respect to claims 16-18.

3. Claims 9, 10, 21, 28, 29, and 50 are unpatentable under 35 U.S.C. 103(a) over St. John Herbert, III (U.S. Patent No. 6,366,917) in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624) and Goldberg et al. (U.S. Patent No. 6,076,092).

Paragraph 70 on page 19 of the Final Official Action rejected claims 9, 10, 21, 28, 29 and 50 under 35 U.S.C. 103(a) as being unpatentable over St. John Herbert, III (U.S. Patent No.

6,366,917) in view of Schmuck et al. (U.S. Patent No. 5,940,841), and further in view of Teare et al. (U.S. Patent No. 6,151,624) and Goldberg et al. (U.S. Patent No. 6,076,092). Appellants respectfully traverse.

Paragraph 71 on page 19 of the Final Official Action rejected claim 9 for similar reasons as stated above with respect to claim 1. Paragraph 71 further said: "St. John Herbert, Schmuck, and Teare do not specifically disclose the client sends the data access command to the data storage device over a data transmission path that bypasses the data mover computer. Goldberg discloses the client sends the data access command to the data storage device over a data transmission path that bypasses the data mover computer [col 10, lines 43-49]. It would have been obvious to a person skill[ed] in the art at the time the invention was made to combine the teachings of St. John Herbert, Schmuck, Teare and Goldberg because Goldberg's teaching of transmission path that bypasses the data mover computer would reduce communication traffic and increase performance for the system."

Goldberg col. 10, lines 43-49 say:

> Since query objects are generated automatically, clients that rely on them can take advantage of emerging technological improvements by simply regenerating their query objects, thereby preserving the IDL interface but upgrading the implementation. Finally, query objects bypass the data object layer, thereby avoiding the need to map data into active data objects.

It is not entirely clear from this passage what the data object layer is. However, there is nothing in Goldberg suggesting that the data object layer is a data mover computer of a file

server or even any kind of layer that would perform a file server function such as mapping logical addresses in a file to data storage locations. Instead, the only other passage in Goldberg where the term "data object layer" is found is column 2, lines 20-23, which relates to a three-tier model for interfacing a database application to a database management system server (DBMS). "Each database object is implemented as a middle tier server that passes an API for the DBMS from the DBMS server to the database object. The clients pass queries into the database object via the API and receive the results sets back. This approach provides the full power of the DBMS query engine to the applications, but provides no abstraction from the query engine." Goldberg, col. 1, lines 52-58. With respect to Goldberg's FIG. 2, Goldberg, in col. 4, lines 34-42, further says: "The database object 33 logically maps the entire database 32 into a single object with a general interface that functions as an abstract API 37 for the underlying database engine (not shown). Clients of the database object 33, in this case, the application 30, can pass arbitrary queries 34 to the database object 33 and obtain the resultant data 35. The clients need not know what query is actually being issued to the database 21."

Apparently the database object layer in Goldberg refers to the middle layer of the three-tiered model. Thus, it refers to the database object 33 in FIG. 2 of Goldberg, and a comparison of FIG. 2 to FIG. 4 of Goldberg suggests that "bypassed" as used in Goldberg means that the database object layer has been replaced with business objects and query objects.

Goldberg as a whole does not appear to be pertinent to a client sending data access commands to a data storage device over a data transmission path that bypasses a data mover computer in a file server. If a file system layer were used in the system of Goldberg et al., it would not be bypassed. Instead, it would be a layer in the lower tier, between the DBMS 55a,b

and the database 54a,b in FIG. 4, for example, if the data source were a flat file as mentioned in col. 6, lines 63-67. Moreover, Goldberg does not bypass the DBMS 55a,b in FIG. 4. Furthermore, Goldberg replaces the database object layer with business objects 83 and query objects 52a, 52b.

In short, Goldberg fails to disclose a client that sends the data access command to a data storage device over a data transmission path that bypasses a data mover computer. Nor is there any proper motivation for combining Goldberg with St. John Herbert, Schmuck, and Teare, in order to reconstruct appellants' invention.

Paragraph 72 on page 20 of the Final Official Action rejected claim 10 for similar reasons as stated above in claim 2. Appellants respectfully traverse, for the reasons as stated above with respect to claim 2.

Paragraph 73 on page 20 of the Final Official Action rejected claim 21 for similar reasons as stated above with respect to claims 1 and 9. Appellants respectfully traverse, for the reasons as stated above with respect to claims 1 and 9.

Paragraph 74 on page 20 of the Final Official Action rejected claims 28 and 29 for similar reasons as stated above with respect to claims 9 and 10. Appellants respectfully traverse, for the reasons as stated above with respect to claims 9 and 10.

Paragraph 75 on page 20 of the Final Official Action rejected claim 50 for similar reasons as stated above in claims 1, 2, 8 and 9. Appellants respectfully traverse, for the reasons as stated above with respect to claims 1, 2, 8 and 9. In addition, claim 50 further defines that the metadata of the file (sent by the data mover to the client) includes "information specifying data storage locations in the cached disk array for storing data of the file," and the client sends the data access

command ("specifying the data storage locations in the cached disk array for storing the data to be written") over "a data path that bypasses the data mover computer to access the data storage locations in the cached disk array for storing the data to be written ...." These limitations of claim 50 further define a kind of use of metadata that is not disclosed in St. John Herbert, III or Teare, and a kind of bypassing for data access that is not disclosed in Goldberg.

Conclusion

1. With respect to all of the independent claims, the cited references fail to show a file server responding to a user or client request for metadata about a file or a request for access to a file by returning metadata including information specifying data storage locations in the file server for storing data of the file or metadata of the file including information specifying data storage locations in a cached disk array for storing data of the file. A file name does not specify data storage locations in the file server for storing data of the file.

2. With respect to all of the independent claims, the hindsight nature of the proposed combination is evident from the fact that the cited references are dealing with four different kinds of data processing systems (database server, video file server, multiprocessor system, and a name finding service in the World Wide Web) and various different kinds of metadata. There is no apparent reason why a person of ordinary skill, seeking to improve the performance of a network file server, would be looking at such disparate references and more importantly selecting the particular portions of the different systems in these references for combination and modification to reconstruct the appellants' claimed invention.

3. With respect to the dependent claims concerning the caching of metadata, Schmuck's metadata caching occurs in a multi-processor system for cache coherency, and not between a network client and a data mover or file server for the network client. Moreover, although Schmuck discloses that a version number is associated with each metadata update, Schmuck fails to show a client sending a metadata request including a version identifier so that the data mover or file server may compare the version identifier in the request to the version identifier of the most recent version of metadata that the data mover or file server has in order to return the most recent version of the metadata to the client when the client does not have the most recent metadata of the file.

4. With respect to the independent claims 21 and 50 and the dependent claims concerning the bypassing of a data mover computer of a file server, the data object layer in Goldberg is not a data mover computer or any kind of layer that would perform a file server function such as mapping logical addresses in a file to data storage locations. Goldberg does not concern a file server or bypassing a data mover computer and instead relates to a three-tier model for interfacing a database application to a database management system server (DBMS).

In view of the above, the rejection of claims 1-50 should be reversed.

Respectfully submitted,

8 April 2004
date

Richard C. Auchterlonie
Reg. No. 30,607
HOWREY SIMON ARNOLD & WHITE, LLP
P.O. Box 4433
Houston, Texas 77210
(713) 787-1400

IX. APPENDIX

The claims involved in this appeal are as follows:

1. A method of operating a file server in a data network, said method comprising:

(a) the file server receiving a request for metadata about a file to be accessed, the request being received from a data processing device in the data network; and

(b) in response to the request for metadata, the file server granting to the data processing device a lock on at least a portion of the file, and returning to the data processing device metadata of the file including information specifying data storage locations in the file server for storing data of the file.

2. The method as claimed in claim 1, wherein the file server includes a data storage device including the data storage locations, and a data mover computer for managing locks on files having data stored in said data storage device, wherein the data storage device stores metadata of a plurality of files having file data stored in the data storage device, the data mover computer is coupled to the data storage device for transfer of the metadata between the data storage device and the data mover computer, the data mover computer has a random access memory, and the method includes the data mover computer maintaining a metadata cache in the random access memory, and the method includes the data mover computer accessing the metadata cache for obtaining the metadata that is returned to the data processing device.

3. The method as claimed in claim 1, wherein a plurality of data processing devices in the data network share read-write access to the file, and the file server grants respective read locks and write locks to the data processing devices in the data network.

4. The method as claimed in claim 1, wherein the data processing device writes data to the data storage locations in the file server, modifies the metadata from the file server in accordance with the data storage locations in the file server to which the data is written, and sends the modified metadata to the file server.

5. The method as claimed in claim 4, wherein the data processing device sends the modified metadata to the file server after the data processing device writes the data to the data storage of the file server.

6. The method as claimed in claim 1, wherein the data processing device has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein the data processing device includes the version identifier in the request for access to the file, the file server compares the version identifier from the data processing device to a version identifier of a most recent version of the metadata of the file, and the file server returns the most recent version of the metadata of the file to the data processing device when the comparison of the version identifier from the data processing device to the version identifier of the most recent version of the metadata of the file indicates that the metadata

of the file cached in the cache memory of the data processing device is not the most recent metadata of the file.

7. The method as claimed in claim 6, wherein the version identifier is a number that is incremented when the metadata of the file is modified.

8. A method of operating a file server and a client in a data network, said method comprising:

(a) the client sending to the file server at least one request for access to a file;

(b) the file server receiving said at least one request for access to the file, granting to the client a lock on at least a portion of the file, and sending to the client metadata of the file including information specifying data storage locations in the file server for storing data of the file;

(c) the client receiving from the file server the metadata of the file, using the metadata of the file to produce at least one data access command for accessing the data storage locations in the file server, and sending the data access command to the file server to access the data storage locations in the file server; and

(d) the file server responding to the data access command by accessing the data storage locations in the file server.

9. The method as claimed in claim 8, wherein the file server includes a data storage device including the data storage locations, and a data mover computer for managing locks on

files having data stored in said data storage device, and wherein the client sends to the data mover computer said at least one request for access to the file, the data mover computer responds to said at least one request for access to the file by returning to the client the metadata of the file, and wherein the client sends the data access command to the data storage device over a data transmission path that bypasses the data mover computer.

10. The method as claimed in claim 9, wherein the data storage device stores metadata of a plurality of files having file data stored in the data storage device, the data mover computer is coupled to the data storage device for transfer of the metadata between the data storage device and the data mover computer, the data mover computer has a random access memory, and the method includes the data mover computer maintaining a metadata cache in the random access memory, and the method includes the data mover computer accessing the metadata cache for obtaining the metadata that is sent to the client.

11. The method as claimed in claim 8, wherein a plurality of clients in the data network share read-write access to the file, and the file server grants respective read locks and write locks to the clients in the data network.

12. The method as claimed in claim 8, wherein the lock on at least a portion of the file granted by the file server to the client is not granted to any particular application process of the client, and wherein the client has a lock manager that grants a local file lock to a particular application process that accesses the file.

13. The method as claimed in claim 8, wherein the client has a lock manager that responds to a request from an application process of the client for access to the file by granting to the application process a local file lock on at least a portion of the file, and then sending to the file server said at least one request for access to the file.

14. The method as claimed in claim 8, wherein the method includes dynamically linking application programs of the client with input-output related operating system routines of the client, the input-output related operating system routines intercepting file access calls from client application processes to send file access requests to the file server to obtain from the file server locks upon at least a portion of each of the files, to obtain metadata for producing data access commands for accessing data storage in the file server, to produce the data access commands from the metadata, and to send the data access commands to the file server in order to access the data storage of the file server.

15. The method as claimed in claim 8, wherein the data access command is a write command for a write operation upon at least a portion of the file, and wherein the method includes the client writing the data to the data storage locations in the file server, modifying the metadata from the file server in accordance with the write operation upon at least a portion of the file, and sending the modified metadata to the file server.

16. The method as claimed in claim 15, wherein the client sends the modified metadata to the file server after the client writes the data to the data storage of the file server.

17. The method as claimed in claim 16, wherein the client performs asynchronous write operations upon the data storage locations of the file server, and wherein the client sends the modified metadata to the file server in response to a commit request from an application process of the client.

18. The method as claimed in claim 16, wherein the client performs asynchronous write operations upon the data storage locations of the file server, and wherein the client sends the modified metadata to the file server when the client requests the file server to close the file.

19. The method as claimed in claim 8, wherein the client has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein the client includes the version identifier in the request for access to the file, the file server compares the version identifier from the client to a version identifier of a most recent version of the metadata of the file, and the file server returns the most recent version of the metadata of the file to the client when the comparison of the version identifier from the client to the version identifier of the most recent version of the metadata of the file indicates that the metadata of the file cached in the cache memory of the client is not the most recent metadata of the file.

20. The method as claimed in claim 19, wherein the version identifier is a number that is incremented when the metadata of the file is modified.

21. A file server comprising:

at least one data storage device for storing a file system; and

a data mover computer coupled to the data storage device for exchange of metadata of files in the file system, the data mover computer having at least one network port for exchange of control information and metadata of files in the file system with data processing devices in the data network, the control information including metadata requests;

wherein the data storage device has at least one network port for exchange of data with the data processing devices in the data network over at least one data path that bypasses the data mover computer; and

wherein the data mover computer is programmed for responding to each metadata request for metadata of a file from each data processing device by granting to said each data processing device a lock on at least a portion of the file, and returning to said each data processing device metadata of the file including information specifying data storage locations in the data storage device for storing data of the file.

22. The file server as claimed in claim 21, wherein the data mover computer is programmed to receive modified metadata from said each data processing device, and write the modified metadata to the data storage device.

23. The file server as claimed in claim 21, wherein the data mover computer has a random access memory, and the data mover computer is programmed for maintaining a metadata cache in the random access memory, and the data mover computer is programmed for accessing the metadata cache for obtaining the metadata that is returned to said each data processing device.

24. The file server as claimed in claim 23, wherein the data mover computer is programmed for receiving modified metadata from said each data processing device, and writing the modified metadata to the metadata cache in the random access memory.

25. The file server as claimed in claim 21, wherein the data mover computer is programmed for receiving a metadata version identifier in the metadata request to a version identifier of a most recent version of the metadata of the file, and for returning the most recent version of the metadata of the file to said each data processing device when the comparison indicates that the metadata version identifier in the each metadata request fails to identify the most recent version of the metadata of the file.

26. The file server as claimed in claim 25, wherein the version identifier is a number, and the data mover computer is programmed to increment the version identifier when the metadata of the file is modified.

27. A data processing system comprising, in combination;

a file server; and

a plurality of clients linked by a data network to the file server;

wherein the file server is programmed for receiving from each client at least one request for access to a file, for granting to said each client a lock on at least a portion of the file, and for sending to said each client metadata of the file including information specifying data storage locations in the file server for storing data of the file;

wherein said each client is programmed for using the metadata of the file to produce at least one data access command for accessing data of the file; and

wherein the file server is programmed for receiving from said each client said at least one data access command for accessing data of the file by accessing the data storage locations in the file server.

28. The data processing system as claimed in claim 27, wherein the file server includes a data storage device including the data storage locations, and a data mover computer programmed for managing locks on files having data stored in said data storage device, wherein the data mover computer has a network port for receipt of file access requests form clients, and wherein the data storage device has a network port for receipt of data access commands from said clients over at least one data transmission path that bypasses the data mover computer.

29. The data processing system as claimed in claim 28, wherein the data storage device stores metadata of a plurality of files having file data stored in the data storage device, the data mover computer is coupled to the data storage device for the transfer of the metadata

between the data storage device and the data mover computer, the data mover computer has a random access memory, and the data mover computer is programmed for maintaining a metadata cache in the random access memory, and for accessing the metadata cache for obtaining the metadata that is sent to said each client.

30. The data processing system as claimed in claim 27, wherein a plurality of clients in the data network share read-write access to files stored in data storage of the file server, and the file server is programmed to grant respective read locks and write locks to the clients in the data network.

31. The data processing system as claimed in claim 27, wherein the lock on at least a portion of the file granted by the file server to the client is not granted to any particular application process of said each client, and wherein said each client has a lock manager for granting a local file lock to a particular application process that accesses the file.

32. The data processing system as claimed in claim 27, wherein said each client has a lock manager for responding to a request from an application process of said each client for access to the file by granting to the application process a local file lock on at least a portion of the file, and then sending to the file server said at least one request for access to the file.

33. The data processing system as claimed in claim 27, wherein said client is programmed with input-output related operating system routines for intercepting file access calls from client application processes for sending file access requests to the file server.

34. The data processing system as claimed in claim 33, wherein the data access commands include at least one write command for a write operation upon at least a portion of at least one file, and wherein the client is programmed for writing data to data storage locations in the file server, modifying the metadata from the file server in accordance with the write operation upon at least a portion of said at least one file, and sending the modified metadata to the file server.

35. The data processing system as claimed in claim 27, wherein said each client has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein said each client is programmed to include the version identifier in the request for access to the file, the file server is programmed to compare the version identifier from said each client to a version identifier of a most recent version of the metadata of the file, and the file server is programmed to return the most recent version of the metadata of the file to the client when the comparison of the version identifier from said each client to the version identifier of the most recent version of the metadata of the file indicates that the metadata of the file cached in the cache memory of said each client is not the most recent metadata of the file.

36. A program storage device containing a program for a file server, the file server having at least one data storage device for storing a file system, and having at least one network port for exchange of control information and metadata of files in the file system with at least one data processing device, the control information including metadata requests, wherein the program is executable by the file server for responding to each metadata request for metadata of a file by granting to said each data processing device a lock on at least a portion of the file, and returning to said each data processing device metadata of the file including information specifying data storage locations in the data storage device for storing data of the file.

37. The program storage device as claimed in claim 36, wherein the program is executable by the file server for receiving modified metadata from the data processing device and writing the modified metadata to data storage of the file server.

38. The program storage device as claimed in claim 36,

wherein the file server has nonvolatile data storage containing the file system and metadata of files in the file system, and the file server has a random access memory; and

wherein the program is executable by the file server for maintaining a metadata cache in the random access memory, and for accessing the metadata cache for obtaining the metadata that is returned to the data processing device.

39. The program storage device as claimed in claim 38, wherein the program is executable by the file server for receiving modified metadata from the data processing device, and writing the modified metadata to the metadata cache in the random access memory.

40. The program storage device as claimed in claim 38, wherein the program is executable by the file server for receiving a metadata version identifier in the metadata request to a version identifier of a most recent version of the metadata of the file, and for returning the most recent version of the metadata of the file to the data processing device when the comparison indicates that the metadata version identifier in the metadata request fails to identify the most recent version of the metadata of the file.

41. The program storage device as claimed in claim 40, wherein the version identifier is a number, and the program is executable by the file server for incrementing the version identifier when the metadata of the file is modified.

42. A program storage device containing a program for a data processing device that is a client in a data network, the program being executable by the client to enable application programs of the client to access files in data storage of at least one file server in the data network, the program being executable in response to a call from an application program for access to data of a file by sending to the file server a metadata request for metadata of the file including information specifying data storage locations for data of the file in the file server, receiving the metadata of the file from the file server, using the metadata of the file to produce at least one data

access command for accessing the data storage locations in the file server, and sending the data access command to the file server to access the data storage locations in the file server.

43. The program storage device as claimed in claim 42, wherein the program is executable by the client in response to the call from the application program by granting a local file lock to a particular application process that is executing the application program, and then sending to the file server the metadata request for metadata of the file.

44. The program storage device as claimed in claim 42, wherein the program includes input-output related operating system routines for intercepting file access calls from the client application programs.

45. The program storage device as claimed in claim 42, wherein the data access command is a write command for a write operation upon at least a portion of the file, and wherein the program is executable by the client for writing the data to the data storage locations in the file server, modifying the metadata from the file server in accordance with the write operation upon at least a portion of the file, and sending the modified metadata to the file server.

46. The program storage device as claimed in claim 45, wherein the program is executable by the client for sending the modified metadata to the file server after the client writes the data to the data storage of the file server.

47. The program storage device as claimed in claim 45, wherein the program is executable for sending the modified metadata to the file server in response to a commit request from the application program.

48. The program storage device as claimed in claim 45, wherein the program is executable for sending the modified metadata to the file server when the client requests the file server to close the file.

49. The program storage device as claimed in claim 42, wherein the client has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein the program is executable by the client for including the version identifier in the metadata request that is sent to the file server.

50. A method of operating a file server and a client in a data network, the file server having a cached disk array including data storage locations, and a data mover computer for managing locks on files having data stored in the cached disk array, said method comprising:

(a) the client sending to the data mover computer at least one request for write access to a file;

(b) the data mover computer receiving said at least one request for write access to the file, granting to the client a lock on at least a portion of the file, and sending to the client metadata of the file including information specifying data storage locations in the cached disk array for storing data of the file;

(c) the client receiving from the data mover computer the metadata of the file, using the metadata of the file to produce at least one data access command for writing data to the data storage locations in the cached disk array for storing data of the file, the data access command including the data to be written to the data storage locations in the cached disk array for storing data of the file and specifying the data storage locations in the cached disk array for storing the data to be written, and sending the data access command over a data path that bypasses the data mover computer to access the data storage locations in the cached disk array for storing the data to be written;

(d) the file server responding to the data access command by writing the data to be written to the data storage locations in the cached disk array for storing the data to be written;

(e) the client modifying the metadata from the data mover computer in accordance with the writing of the data to be written to the data storage locations in the cached disk array for storing the data to be written; and

(f) the client sending the modified metadata to the data mover computer after the data has been written to the data storage locations in the cached disk array for storing the data to be written.